

ICASE REPORT

(NASA-CR-142599) ACCESSING TECHNICAL DATA
BASES USING STDS: A COLLECTION OF SCENARIOS
(Universities Space Research Association)

CSCD 09B

N75-21042

Unclas

G3/61 18578

ACCESSING TECHNICAL DATA BASES USING STDS:

A COLLECTION OF SCENARIOS

W. T. Hardgrave

Report Number 75-8

April 16, 1975

INSTITUTE FOR COMPUTER APPLICATIONS

IN SCIENCE AND ENGINEERING

Operated by the

UNIVERSITIES SPACE RESEARCH ASSOCIATION

at

NASA'S LANGLEY RESEARCH CENTER

Hampton, Virginia

PRICES SUBJECT TO CHANGE

Reproduced by
**NATIONAL TECHNICAL
INFORMATION SERVICE**
U.S. Department of Commerce
Springfield, VA. 22151

N O T I C E

**THIS DOCUMENT HAS BEEN REPRODUCED FROM THE
BEST COPY FURNISHED US BY THE SPONSORING
AGENCY. ALTHOUGH IT IS RECOGNIZED THAT CER-
TAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RE-
LEASED IN THE INTEREST OF MAKING AVAILABLE
AS MUCH INFORMATION AS POSSIBLE.**

ACCESSING TECHNICAL DATA BASES USING STDS:

A COLLECTION OF SCENARIOS

W. T. Hardgrave

ABSTRACT

This report gives a line by line description of sessions using the Set-Theoretic Data System (STDS) to interact with technical data bases. The data bases contain data from actual applications at NASA Langley Research Center. The report is meant to be a tutorial document that accompanies "Set Processing in a Network Environment" [1].

This paper was a result of work performed under NASA Grants NGR 47-102 001 and NSG 1068 while the author was in residence at ICASE, NASA Langley Research Center, Hampton, Virginia 23665.

1.0 Introductory Remarks

This report is directed to the user who has a collection of data that he wants to query and/or update but who has little, if any, knowledge of either set theory or information systems. In only a few sections do we make the tacit assumption that the reader has any experience in computing and these cases involve only a basic knowledge of FORTRAN. We hope to demonstrate, through the use of three scenarios, the power of a set-theoretic system in aiding the user to isolate and examine sets of data that may be of interest to him. The particular system used in this report is the Set-Theoretic Data System (STDS)[2] developed by David Childs and supported by Set Theoretic Information Systems (STIS) Incorporated. STDS was run (remotely from a teletype in Hampton, Virginia) on the IBM 360/67 at Wayne State University, Detroit, Michigan, using Michigan Terminal System (MPS)[3] developed at the University of Michigan. Since STDS is a research system rather than a production system, it is reasonable to expect it to have a number of deficiencies. Few of these can be classified as serious problems; most reside in the area of human factors and although revisions here are desirable, they are not absolutely necessary to solve the problems described herein.

A standard strategy that pervades all of the scenarios, regardless of the data base is as follows. First, isolate the basic sets that can be requested by a single condition (e.g. velocity \leq 100 MPH). Then, using the set operations, isolate sets that have characteristics that are of particular interest to the interrogator. Whenever a new set is formed, the system declares the cardinality (i.e. the number of elements) of the

set. If the user desires more statistical information, he may request a summary of pertinent items that includes the maximums, minimums, means and standard deviations. If the user requires more specialized information about a given set, he may write a short FORTRAN routine that traverses a set, performs calculations and outputs the necessary values. An example of this usage is given in the VGH scenario.

2.0 An Introduction to STDS

In this section, a brief introduction to STDS [2] is given. This consists of two subsections. Subsection 2.1 provides a description of the commands necessary and responses encountered when using STDS. In addition, other aspects of STDS that are common to all of the scenarios (e.g. timings) are dealt with in this subsection.

Subsection 2.2 contains brief descriptions of some elementary STDS operations. Since STDS is primarily a research system maintained by STIS Corporation, it has a number of operations that are not particularly germane to the discussions considered herein. As a result, we list here only the operations that are relevant for a relational/set-theoretic system that might appear in a scientific user environment. These operations are listed in three categories: utility operations, restriction operations, and set operations. Subsection 2.2 is primarily meant to be used as a reference to accompany the examination of the scenarios.

Finally, we offer a few opinions concerning the possible installation of STDS on CDC 6000 machines under KRONOS. STDS is written primarily in FORTRAN with some critical routines written in assembly code. The original STDS was implemented on an IBM 360/67 under MTS and a later implementation is available for IBM 360/370 machines under OS. Our estimate is that STDS could be installed under KRONOS in about three months. No special peripheral processor routines should be necessary. The areas that might cause difficulty are program arrangement and Input/Output (I/O) management. The IBM 360/67 is a virtual memory machine and the original STDS implementation used

that feature to minimize manual segmentation of the program and data areas. Therefore some effort may be necessary in order to reduce the field length of STDS under KRONOS to an acceptable level. This will involve the use of buffer management and overlaying techniques. The I/O philosophies of IBM and CDC are radically different. As a result the conversion of STDS I/O to KRONOS may be tricky. However calls to CIO using the direct access capability should be sufficient. Higher level CDC supplied I/O routines should be avoided.

2.1 STDS Operations

The STDS operations are listed below in three groups; utility operations restriction operations, and set operations. It is necessary to provide some informal definitions of the terms, elements, domains, and entries. Consider the relation

$$A = \{ \langle 1, 2, 3, 4 \rangle, \langle 5, 6, 7, 8 \rangle, \langle 9, 6, 8, 7 \rangle \}$$

which may also be written

A:	1	2	3	4
	5	6	7	8
	9	6	8	7

Informally, the columns are equivalent to domains, the rows to entries. A relation is a set of entries. Each entry is an n-tuple containing n elements. Thus the relation may be represented as a table of elements with the columns corresponding to domains and the rows to entries.

2.1.1 Utility Operations

Name: GET

Calling Sequences:

STDS*: GET(C, 'DSN')

FORTTRAN: CALL GET (C, 'DSN')

Arguments:

C is a set name

'DSN' is a data set name

Description:

A set stored off-line on disk, tape, datacell, etc. is brought into core as set C for active inquiry. The contents of 'DSN' must in fact be a set, not an array. Any 'DSN' for which GET is used must previously have been stored by PUT.

Name: PUT

Calling Sequences:

STDS*: PUT(A,'DSN')

FORTTRAN: CALL PUT(A,'DSN')

Arguments:

A is a set name

'DSN' is a dataset name

Description:

Any set A that has been created during a run using set operations may be saved off-line in direct access storage by use of PUT. The command PUT must have been used to place a set in storage before that set can be retrieved by means of the command GET.

Name: ENTER

Calling Sequences:

STDS*: ENTER(A,DA,NDA,'FMT')

Arguments:

A is the resulting set

DA is the domain declaration for A

NDA is the number of domains in A

FMT is the I/O FORMAT for A

Description:

ENTER allows a set, A, to be constructed by specifying the individual entries. ENTER is terminated by an END OF FILE.

Name: LIST

Calling Sequences:

STDS*: LIST(A,I,J,'FMT')

FORTTRAN: CALL LIST(A,I,J,'FMT')

Arguments:

A is the set to be listed

I,J are respectively the I-th and J-th entries of A

'FMT' is a string name specifying the output format

Description:

This operation allows listing subsets of a set A under a format 'FMT'. The subset is the I-th through the J-th entries of the set A.

Name: LISTU

Calling Sequences:

STDS*: LISTU

Arguments:

None

Description:

This operation allows a summary of the characteristics of all sets in the universe to be printed.

Name: FREE

Calling Sequences:

STDS*: FREE(A)

FORTTRAN: CALL FREE(A)

Arguments:

A is the set to be freed

Description:

FREE(A) deletes A from the universe and frees the memory of its contents.

Name: SETFMT

Calling Sequences:

STDS*: SETFMT(DD,'FMT')

FORTTRAN: I = SETFMT(DD,'FMT')

Arguments:

DD is the domain declaration 8, 16, 32

'FMT' is a FORTTRAN format specification

I is the index associated with the specified format

Description:

SETFMT associates an index, I, with a format so that the format may be specified via '/I/'.

ORIGINAL PAGE IS
OF POOR QUALITY

Name: SYSTEM

Calling Sequences:

STDS*: SYSTEM

FORTRAN: CALL SYSTEM .

Arguments:

None

Description:

SYSTEM returns control to the command system in the host operating system.

Name: SUM

Calling Sequences:

STDS*: SUM(A,INDEX,STAT,F)

FORTRAN: CALL SUM(A,INDEX,STAT,F)

Arguments:

A is the set to be summarized

INDEX is the index set as described below

STAT is the statistical set (NUMD=6) to be generated

F is the floating point indicator; 0 implies integer arithmetic; 1 implies floating point arithmetic

Description:

SUM produces a statistical summary of specified elements.

Index Set:

The index set should be DD = 32 NUMD = 3. Each triple $\langle x,y,z \rangle$ in the index set should have the following form:

X is an integer key. This key is necessary when a number of statistics are being gathered with a single index set.

Y is the starting byte of the domain. For 32 bit values, the starting byte for a domain I can be calculated as follows:

$$Y = 4*(I-1) + 1$$

Z is the length in bytes of the domain. For 32 bit values,

$$Z = 4.$$

Example:

CALL SUM(A,INDEX,STAT,1) using 32-bit values and with
INDEX = $\{ \langle 1,5,4 \rangle, \langle 2,9,4 \rangle \}$ would produce the sum, mean,
minimum, maximum and standard deviation for domains 2 and 3
(with keys 1 and 2 so the user can tell which is which).

Name: V\$

Calling Sequence:

STDS: V\$(\$X,DD,V)

Arguments:

\$X is the data mode. I implies integer, Z implies hexadecimal, A implies EDCDIC coded, and E implies floating point.

DD is the domain declaration (usually 32)

V is the name to be assigned to the constant

Description:

V\$ provides the user with the capability to predefine constants to be used at a later time. After entering the V command, the system responds with a ':'. Then the user enters the constant.

Example:

V\$(\$I,32,KON)

:57

The user enters the V\$ command. The system responds with a ':'. The user enters the integer, '57'. Thus the user creates the integer constant '57' with the alias KON. Whenever KON is encountered in a command, STDS substitutes 57.

2.1.2 Restriction Operations

Name: RSEG, RSLTA, RSLEA, RSGEA, RSGTA, RSNE

Calling Sequences:

STDS#: RS ____ (I,A,V,C)

FORTRAN: CALL RS ____ (I,A,V,C)

Arguments:

I is a domain name

A is the set to be restricted

V is a full word program variable

C is the result set, a subset of A

Description:

The RS ____ commands restrict A (forming C) as dictated by the domain I and the constant V. C is the subset of A containing entries of A whose Ith domain is in the specified relation (e.g. EQ denotes 'equal') to V.

Examples:

For $A = \{ \langle -10, 20, 3 \rangle, \langle 0, 5, 10 \rangle, \langle 6, 7, 1000 \rangle \}$

- a) RSLTA(1,A,-2,C) produces a set $C = \{ \langle -10, 20, 3 \rangle \}$
- b) RSLEA(2,A,10,C) produces a set $C = \{ \langle 0, 5, 10 \rangle, \langle 6, 7, 1000 \rangle \}$
- c) RSGTA(2,A,10,C) produces a set $C = \{ \langle -10, 20, 3 \rangle \}$
- d) RSGEA(3,A,1000,C) produces a set $C = \{ \langle 6, 7, 1000 \rangle \}$
- e) RSNE(1,A,10,C) produces a set $C = A$

ORIGINAL PAGE IS
OF POOR QUALITY

Name: BRSA

Calling Sequences:

STDS*: BRSA(I,A,J,K,C)

FORTTRAN: CALL BRSA(I,A,J,K,C)

Arguments:

I is a domain name

A is the set to be restricted

J,K are four byte program variables

C is the result set, a subset of A

Description:

The set C is a subset of the set A such that the values of the entries in the I-th domain of C are greater than or equal to J and less than or equal to K. The set A is unchanged.

Examples:

if $A = \{ \langle 1,2,3,4 \rangle, \langle 1,3,5,2 \rangle, \langle 2,3,6,7 \rangle \}$

a) then BRSA(2,A,1,2,C) produces the set

$C = \{ \langle 1,2,3,4 \rangle \}$

b) then BRSA(3,A,4,10,C) produces the set

$C = \{ \langle 1,3,5,2 \rangle, \langle 2,3,6,7 \rangle \}$

c) and BRSA(4,A,4,10,C) produces the set

$C = \{ \langle 1,2,3,4 \rangle, \langle 2,3,6,7 \rangle \}$

Name: ALEG

Calling Sequences:

STDS*: ALEG(A,I,J,LEN,CL,CE,CG)

FORTTRAN: CALL ALEG(A,I,J,LEN,CL,CE,CG)

Arguments:

A is the set to be partitioned

I,J are domains

LEN is the length of the domain in bytes (usually 4)

CL,CE,CG are the output sets

Description:

The set A is partitioned into three sets, CL, CE and CG according to domains I and J. For each entry, K of A, if domain I is less than domain J then the Kth entry is placed in CL; if domain I is greater than domain J, then the Kth entry is placed in CG; and if they are equal, the Kth entry is placed in CE.

Example:

If $A = \{ \langle 1,2,3,4 \rangle, \langle 1,3,5,5 \rangle, \langle 2,3,7,6 \rangle, \langle 5,3,2,7 \rangle \}$

then ALEG (A,3,4,4,CL,CE,CG)

produces $CL = \{ \langle 1,2,3,4 \rangle, \langle 5,3,2,7 \rangle \}$

$CE = \{ \langle 1,3,5,5 \rangle \}$

$CG = \{ \langle 2,3,7,6 \rangle \}$

ORIGINAL PAGE IS
OF POOR QUALITY

Name: DMR

Calling Sequences:

STDS*: DMR(I,J,A,C)

FORTTRAN: CALL DMR(I,J,A,C)

Arguments:

I is a domain name for the set A

J is the number of consecutive domains

A is the given set

C is the result set, the I-th domain of A

Description:

The set C is formed from the set A by taking the I-th through J-th domains of each entry of the set A. The set A is unchanged.

Examples:

For A = {< a,b,c >, < d,e,f >, < g,h,i >}

DMR(1,2,A,C) produces

C = {< a,b >, < d,e >, < g,h >}

Name: GETELM

Calling Sequences:

FORTTRAN: CALL GETELM(I,A,J,L,ELM)

Arguments:

I is the entry number

A is the set name

J is the first byte of the tuple to be extracted

L is the number of bytes to be extracted (usually 4)

ELM is an array

Description:

GETELM extracts L bytes beginning at the Jth byte from the Ith entry of the set A. This information is transferred to the user's array ELM. For 32-bit values (4 bytes) the byte number may be calculated from the domain number by the formula $BN = 4 \times DN - 3$ where BN is the byte number and DN is the domain number.

Examples: (using 32 bit values)

If $A = \{ \langle 1, 2, 3, 4 \rangle, \langle 1, 3, 5, 2 \rangle, \langle 2, 3, 6, 7 \rangle \}$

a) Then GETELM (2,A,5,4,ARR) produces

ARR(1) = 3

ARR(2) = 5

b) Then GETELM (3,A,1,3,ARR) produces

ARR(1) = 2

ARR(2) = 3

ARR(3) = 6

Name: SUBSET

Calling Sequence:

STDS: SUBSET(A,F,I,L,C)

FORTTRAN: CALL SUBSET(A,F,I,L,C)

Arguments:

A is the input set

C is the output set

F is the first entry

L is the last entry

I is the increment

Description:

The Fth through the Lth elements in increments of I are copied from A thus forming C.

Examples:

For $A = \{ \langle a, b, c \rangle, \langle d, e, f \rangle, \langle g, h, i \rangle \}$

SUBSET(A,1,2,3,C) produces

$C = \{ \langle a, b, c \rangle, \langle g, h, i \rangle \}$

2.1.3 Set Operations

Name: UN

Calling Sequences:

STDS*: UN(AB,C,)

FORTTRAN: CALL UN(A,B,C)

Arguments:

A,B are names of given sets to be unioned together

C is the name of the result set

Description:

Given two sets A and B, a new set C is produced containing each entry that is in either A or B. Note that if the same entry is in both A and B, only one representation of it appears in C. A and B are unchanged.

Examples:

If $A = \{ \langle a,b,c \rangle, \langle d,e,f \rangle, \langle g,h,i \rangle, \langle j,k,l \rangle \}$

and

$B = \{ \langle a,b,c \rangle, \langle m,n,o \rangle, \langle p,q,r \rangle, \langle s,t,u \rangle \}$

then UN(A,B,C) produces

$C = \{ \langle a,b,c \rangle, \langle d,e,f \rangle, \langle g,h,i \rangle, \langle j,k,l \rangle, \langle m,n,o \rangle, \langle p,q,r \rangle, \langle s \}$

Note that $\langle a,b,c \rangle$ is in both A and B, and occurs but once in set C.

Name: IN

Calling Sequences:

STDS*: IN(A,B,C)

FORTTRAN: CALL IN(A,B,C)

Arguments:

A,B are the given sets to be intersected

C is the result set

Description:

The set C contains all those entries that are entries of both the sets A and B. The sets A and B are unchanged.

Examples:

1. For $A = \{ \langle a,b,c \rangle, \langle d,e,f \rangle, \langle g,h,i \rangle \}$

and

$B = \{ \langle a,b,c \rangle, \langle g,h,i \rangle, \langle j,k,l \rangle \}$

IN(A,B,C) produces

$C = \{ \langle a,b,c \rangle, \langle g,h,i \rangle \}$

Name: RL

Calling Sequences:

STDS*: RL(A,B,C)

FORTTRAN: CALL RL(A,B,C)

Arguments:

A,B are the given sets

C is the output set

Description:

The set C contains all those entries of the set A that are not entries of the set B. The sets A and B are unchanged.

Examples:

For $A = \{ \langle a,b,c \rangle, \langle d,e,f \rangle \}$

and

$B = \{ \langle a,b,c \rangle, \langle g,h,i \rangle, \langle j,k,l \rangle \}$

a) RL(A,B,C) produces

$C = \{ \langle d,e,f \rangle \}$ because $\langle d,e,f \rangle$ is not in B

b) RL(B,A,C) produces

$C = \{ \langle g,h,i \rangle, \langle j,k,l \rangle \}$ because $\langle g,h,i \rangle$ and $\langle j,k,l \rangle$ are not in A

ORIGINAL PAGE IS
OF POOR QUALITY

Name: SD

Calling Sequences:

STDS*: SD(A,B,C)

FORTRAN: CALL SD(A,B,C)

Arguments:

A,B are the names of the given sets whose symmetric
difference is to be found

C is the name of the result set

Description:

Given two sets A and B, a set C is produced that is the set of
entries present in either A or B but not in both. That is,
C contains the entries that the two sets do not hold in
common. A and B are unchanged.

Examples:

For A = {< a,b,c >, < d,e,f >, < g,h,i>}

and

B = {< a,b,c >, < d,e,f >, < x,y,z>}

SD(A,B,C) produces

C = {< g,h,i >, < x,y,z>}

Name: CARD

Calling Sequences:

FORTRAN: I = CARD (A)

Arguments:

A is the set

Description:

Returns the cardinality(i.e. the number of elements) of the
set A in location I.

Examples:

If $A = \{ \langle 1, 2, 3, 4 \rangle, \langle 1, 3, 5, 5 \rangle, \langle 2, 3, 7, 6 \rangle \}$

then $I = \text{CARD}(A)$ produces $I = 3$

2.2 Entering STDS

The MTS command `RUN RBLG:STDS*` calls STDS into execution. MTS issues the `EXECUTION BEGINS` message just prior to transferring control to STDS. When STDS becomes active, it generates a heading and a request for the user to enter a universe name. The universe is the set of all sets known to STDS and is of little importance to the user. A list of the sets in the universe may be obtained by using the `LISTU` command. The user enters a four-character universe name of his selection; however, he must take care not to use that name for some other set during the session. Now the user is ready to proceed with his requests.

At this point, the user would normally access some data base (i.e. set) that he has previously saved as a permanent file with the `PUT` command. He may do this with the `GET` command; for example, `GET(A,'FE')` finds the permanent file with the name `FE` and associates it with the local set name, `A`. Next, we discuss the two lines of output that follow the `GET` command, or any other set creation command. Consider some sample lines that might follow a `GET` command; the first,

```
DD = 32  NUMD = 9  CARD = 420
```

indicates that the domain declaration is 32, the number of domains in the set is 9, and the cardinality of the set is 420. The domain declaration of 32 implies that values in the set are stored as 32-bit entities. The number of domains equal to 9 implies that each element of the set is a 9-tuple; or more informally, the set has 9 columns of information. A cardinality of 420 implies that the set has 420 elements. Informally, that means that the set has 420 rows. However, it is important to

remember that theoretically the rows are unordered and no two rows would have identical values in all domains.

The second line printed after a set creation, enclosed in brackets as shown below, contains timing information. All times are in seconds.

[0.005, 0.05 / 0.136, 0.90]

The two numbers preceding the slash indicate STDS operation times; first CPU time and then elapsed (i.e. real) time. The numbers following the slash are total times including MTS overhead; first CPU time and then elapsed time. They will, of course, vary with the load on the computing system.

This same GET command used 0.005 CPU seconds and was completed in 0.05 seconds of real time (i.e. wall clock time). However, the CPU time used including MTS overhead (for virtual memory paging, etc.) was 0.136 seconds and the total real time from receipt of carriage return to the issuance of the first character of output was 0.90 seconds.

3.0 Finite Element Scenario

The following scenario describes a situation that might arise in a computer aided aircraft design environment using a set theoretic information system (in this case STDS). Assume the designer is working with a finite element model of a fuselage, a portion of which is shown in Figure 3-1. Furthermore, assume that the model has been run through a suitable finite element program (in this case, SNAP) applying suitable loading and boundary conditions as shown in Figure 3-2. These input conditions may be represented as relations (because of their tabular nature) and maintained under STDS. We have not chosen to do that in this scenario because our original data was output from (rather than input to) the finite element program SNAP. This results in a data base of forces and moments that the designer wishes to examine in order to evaluate the design. The set-theoretic data base examined in this session has nine domains with 250 entries. A portion of the data base is shown below where FX, FY, FZ represent forces and MX, MY, MZ represent moments.

<u>FE</u>	<u>TYPE</u>	<u>NODE</u>	<u>FX</u>	<u>FY</u>	<u>FZ</u>	<u>MX</u>	<u>MY</u>	<u>MZ</u>
1	Beam	1	78.	-71.	0.	0.	0.	5980.
1	Beam	2	-78.	71.	0.	0.	0.	-9640.
1	Quad	1	-78.	289.	241.	-0.	-0.	-0.
1	Quad	2	-160.	599.	-696.	-0.	-0.	-0.
1	Quad	9	136.	-509.	825.	-0.	-0.	-0.

ORIGINAL PAGE IS
OF POOR QUALITY

27

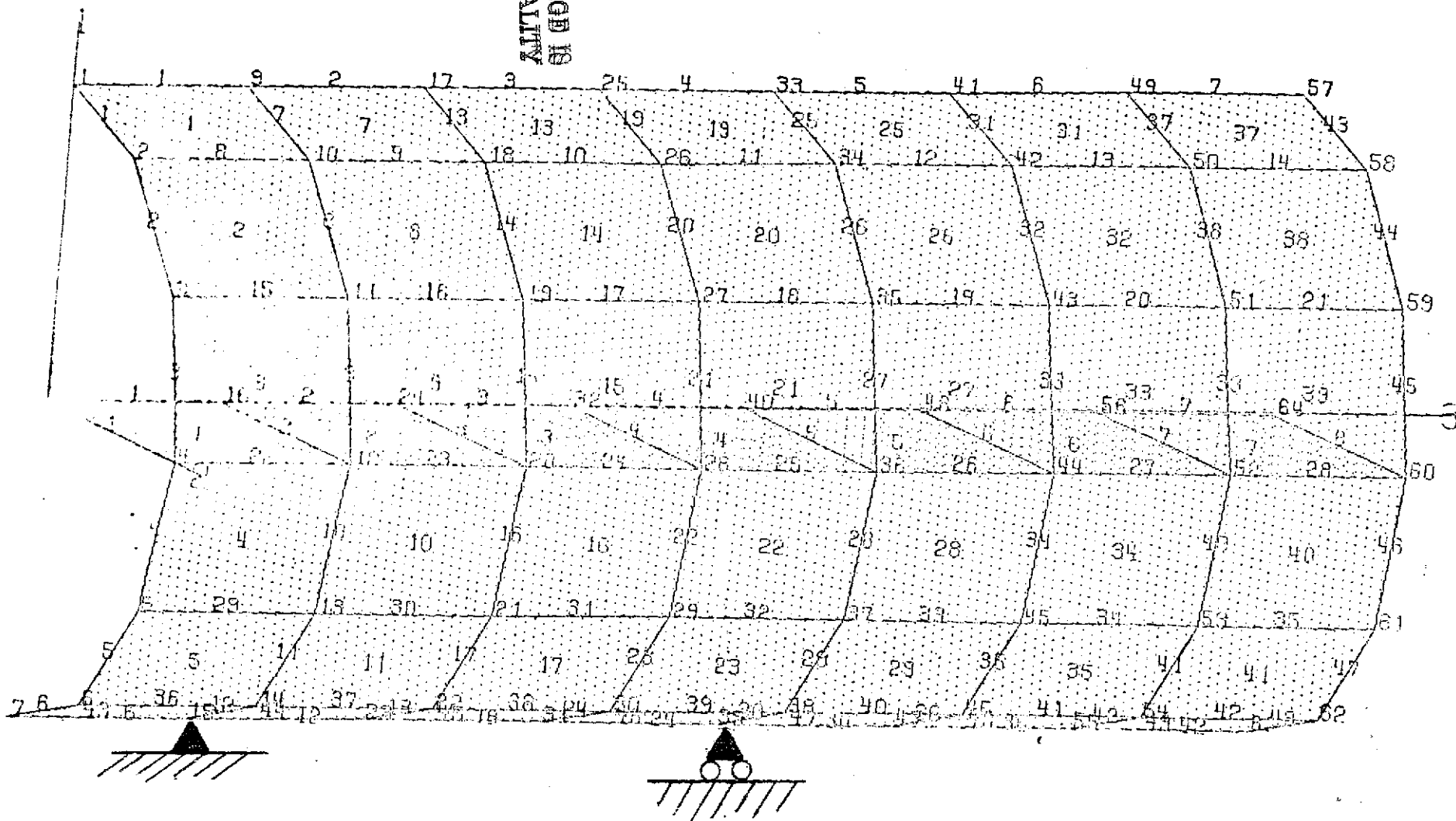


Figure 3-1

BOUNDARY CONDITIONS

<u>NODE</u>	<u>TX</u>	<u>TY</u>	<u>TZ</u>	<u>RX</u>	<u>RY</u>	<u>RZ</u>
1	0	1	0	1	0	1
9	0	1	0	1	0	1
15	1	1	1	1	1	1
17	0	1	0	1	0	1
25	0	1	0	1	0	1
33	0	1	0	1	0	1
39	1	1	0	1	1	1
41	0	1	0	1	0	1
49	0	1	0	1	0	1
57	0	1	0	1	0	1

0 - FREE TO MOVE

1 - FIXED

LOAD CONDITIONS

<u>NODE</u>	<u>DIRECTION</u>	<u>MAGNITUDE</u>
57	3	10000.
58	3	8666.
59	3	5000.
60	1	-10000.
61	3	-5000.
62	3	-8666.
63	3	-10000.

Figure 3-2

After initializing STDS, as shown in Figure 3-3, we are ready to obtain access to the finite element data base (called FE). This is shown in Figure 3-4. The command GET(FE,'FE') retrieves the FE data base from the MTS permanent file system. The data base had been previously input to STDS and stored in set-theoretic format with the PUT command. In order to get a feeling for the data base, we would like to print a few entries on the terminal. This may be accomplished with the LIST command. However, it is necessary to provide a FORTRAN format to the list command; and since formats are typically long and cumbersome to enter, we may declare it in a SETFMT command. STDS RESPONDS WITH FMT = 1 indicating that for future reference we may access this format with /1/. We list the first few elements of FE using E-format, and then, having discovered the magnitudes involved, we define a new format using the F-specification and list again using the new format, /2/.

In order to obtain statistical summaries for the overall behavior of forces and moments, using the SUM command, it is necessary to define an index set that specifies which domains are to be tabulated. We have pre-defined a suitable index set "FE-INDEX" and stored it in the MTS permanent file system. We may recall that set with GET(INDEX, "FE-INDEX"). Following that we perform a summary on FE. The index set was set up to tabulate domains 4,5,6,7,8,9 (as listed under KEYS) representing FX, FY, FZ, MX, MY and MZ respectively. From this we notice immediately that the highs, lows, and standard deviations indicate that there may be doubtful areas in FX, FY, FZ, and MZ.

```
#RUN RBLG:STDS#  
#EXECUTION BEGINS
```

```
** SET-THEORETIC INFORMATION MANAGEMENT INTERACTIVE INTERFACE **  
[** STDS-I ** VERS:UM02-17-74]
```

```
*** **  
*ENTER 4-CHARACTER UNIVERSE NAME *  
:UNIV  
*UNIVERSE UNIV HAS BEEN CREATED *  
*** **
```

```
"TAU-ON"  
[CORE-MAX(8-PAGE): 16 16 16]
```

Figure 3-3


```

?GET(FE,'FE')
DD=32 NUMD= 9 CARD= 420
[ 0.005, 0.05/ 0.136, 0.901

?SETFMT(32,'(I4,I4,A4,I4,6(1X,E8.1))')
FMT= 1

?LIST(FE,1,5,'/1/')
1 BEAM 1 0.8E 02 -0.7E 02 0.0 0.0 0.0 0.6E 04
1 BEAM 2 -0.8E 02 0.7E 02 0.0 0.0 0.0 -0.1E 05
1 QUAD 1 -0.3E 02 0.3E 03 0.2E 03 -0.0 -0.0 -0.0
1 QUAD 2 -0.2E 03 0.6E 03 -0.7E 03 -0.0 -0.0 -0.0
1 QUAD 9 0.1E 03 -0.5E 03 0.8E 03 -0.0 -0.0 -0.0

?SETFMT(32,'(I4,I4,A4,I4,6(1X,F8.0))')
FMT= 2

?LIST(FE,1,5,'/2/')
1 BEAM 1 78. -71. 0. 0. 0. 5980.
1 BEAM 2 -78. 71. 0. 0. 0. -9640.
1 QUAD 1 -78. 229. 241. -0. -0. -0.
1 QUAD 2 -160. 599. -696. -0. -0. -0.
1 QUAD 9 136. -509. 825. -0. -0. -0.

?GET(INDEX,'FE-INDEX')
DD=32 NUMD= 3 CARD= 6
[ 0.005, 0.04/ 0.145, 0.801

?SUM(FE,INDEX,S1,1)
*** KEY SUM MEAN LOW HIGH STD ***
4 -.142330E 02 -0.03 -12600.00 16400.00 2902.33
5 0.879805E 02 0.21 -47000.00 61200.00 7755.78
6 0.562503E 00 0.00 -91800.00 92900.00 13156.59
7 0.0 0.0 -0.00 0.00 0.00
8 0.0 0.0 -0.00 0.00 0.00
9 -.374224E 06 -391.01 -234000.00 323000.00 42365.67
DD=32 NUMD= 6 CARD= 6
[ 0.525, 1.34/ 0.756, 44.95]

```

Figure 3-4

Assume for this scenario that moments greater than 15000 and forces greater than 1500 are considered excessive and the designer wished to isolate cases where those conditions occurred. As shown in Figure 3-5 we decide to isolate the positive moments in Z. We do this by using the RSGTA command (Restrict-Greater-than-Arithmetic) and forming a new set, POSMZ. A statistical summary reveals that the mean positive MZ is 49028.05 considerably above the tolerable level of 15000.0. In order to determine exactly which entries are excessive, we isolate those that are equal to or exceed the threshold with the BRSA command (Between Restrict - Arithmetic). This command forms a new set EXCESSMZ having those entries whose values in domain 9 are between (and including) 15000.0 and 323000.0. STDS responds that there are 28 of those - more than we would like to examine in detail. Therefore, we isolate the worst ones in EX2. We find there are only 3 between 200000.0 and 323000.0; so we list them and we find that they involve nodes 7, 15 and 61.

In Figure 3-6, we decide to isolate the finite elements that connect to nodes 7, 15, and 61; the worst offenders in positive moments in Z. Using the RSEQ command (Restrict Equal) on domain 3, we form the sets NODE7, NODE15 and NODE61 and list their contents.

We perform a statistical summary on NODE61 in order to check that the forces and moments about node 61 sum to zero and -5000.0 (see figure 3.2). We notice that the sums for FY and FZ are -3.0 and -4998.0 respectively. The designer may wish to examine his loads and boundary conditions to determine if these discrepancies are justified.

?RSGTA(9,FE,0.0,POSMZ)

DD=32 NUMD= 9 CARD= 53

[0.025, 0.03/ 0.162, 1.161

?SUM(POSMZ,INDEX,S2,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
4	0.181670E 05	342.77	-9620.00	9620.00	2545.64
5	0.102513E 05	193.42	-16600.00	16600.00	4446.21
6	-.140380E-09	-0.00	-0.00	0.00	0.00
7	0.325261E-16	0.00	0.00	0.00	0.00
8	-.140238E-03	-0.00	-0.00	0.0	0.00
9	0.259849E 07	49023.05	107.00	323000.00	65361.26

DD=32 NUMD= 6 CARD= 6

[0.071, 0.09/ 0.286, 43.131

?BRSA(9,POSMZ,15000.0,323000.0,EXCESSMZ)

DD=32 NUMD= 9 CARD= 28

[0.004, 0.00/ 0.092, 0.201

?BRSA(9,EXCESSMZ,200000.0,323000.0,EX2)

DD=32 NUMD= 9 CARD= 3

[0.023, 0.00/ 0.091, 0.971

?LIST(EX2,1,3,'/2/')

6 BEAM	7	-4610.	10100.	0.	0.	0.	323000.
12 BEAM	15	-9620.	-16600.	0.	0.	0.	225000.
46 BEAM	61	5210.	7080.	0.	0.	0.	214000.

Figure 3-5

?RSEQ(3,FE,7,NODE7)

DD=32 NUMD= 9 CARD= 3
[0.012, 0.01/ 0.144, 0.40]

?LIST(NODE7,1,3,'/2/')

6 BEAM	7	-4610.	10100.	0.	0.	0.	323000.
6 QUAD	7	4610.	17200.	13000.	-0.	-0.	-0.
99 BEAM	7	0.	0.	-13000.	0.	0.	0.

?RSEQ(3,FE,15,NODE15)

DD=32 NUMD= 9 CARD= 5
[0.012, 0.01/ 0.148, 1.72]

?LIST(NODE15,1,5,'/2/')

6 QUAD	15	-10000.	-37400.	32900.	-0.	-0.	-0.
12 BEAM	15	-9620.	-16600.	0.	0.	0.	225000.
12 QUAD	15	-12600.	-47000.	-37600.	-0.	-0.	-0.
99 BEAM	15	0.	0.	13000.	0.	0.	0.
100 BEAM	15	0.	0.	-8230.	0.	0.	0.

?RSEQ(3,FE,61,NODE61)

DD=32 NUMD= 9 CARD= 5
[0.011, 0.01/ 0.144, 0.33]

?LIST(NODE61,1,5,'/2/')

40 QUAD	61	8200.	2200.	392.	-0.	-0.	-0.
41 QUAD	61	-8790.	-8790.	-7620.	-0.	-0.	-0.
46 BEAM	61	5210.	7030.	0.	0.	0.	214000.
47 BEAM	61	-4620.	-493.	0.	0.	0.	-214000.
91 BEAM	61	0.	0.	1730.	0.	0.	0.

?SUM(NODE61,INDEX,S,1)

KEY	SUM	MEAN	LOW	HIGH	STD
4	0.0	0.0	-8790.00	3200.00	6212.77
5	-300000E 01	-0.60	-8790.00	7030.00	5147.30
6	-499800E 04	-999.60	-7620.00	1730.00	3372.15
7	0.0	0.0	0.0	0.0	-0.0
8	0.0	0.0	0.0	0.0	-0.0
9	0.0	0.0	-214000.00	214000.00	135345.44

DD=32 NUMD= 6 CARD= 6
[0.013, 0.05/ 0.224, 42.34]

Figure 3-6

In Figure 3-7, we isolate the forces in X, Y, and Z that exceed the positive threshold point of 1500.0. Having isolated these in BIGFX, BIGFY and BIGFZ, we would like to find those entries where the threshold is exceeded in all three dimensions. This can be done by intersecting BIGFX and BIGFY forming BIGFXY and then intersecting BIGFZY with BIGFXY forming BIGFXYZ. We discover that only 12 entries exceed in all three dimensions and we list those entries.

Finally in Figure 3-8 some information about timings and costs is provided. The total time spent by the designer in querying the data base was less than 20 minutes (as shown by elapsed time). The CPU time on a IBM 360/67 was less than 20 seconds and the cost of the job was less than \$4.00. Note that the dollar amounts given here are for a university environment and would be somewhat higher in a commercial environment.

```
?BRSA(4, FE,1500.0,16400.0,BIGFX)
DD=32 NUMD= 9 CARD= 53
[ 0.017, 0.02/ 0.154, 1.11]
```

```
?BRSA(5, FE,1500.0,61200.0,BIGFY)
DD=32 NUMD= 9 CARD= 55
[ 0.017, 0.03/ 0.152, 0.22]
```

```
?BRSA(6, FE,1500.0,92900.0,BIGFZ)
DD=32 NUMD= 9 CARD= 103
[ 0.019, 0.02/ 0.154, 0.18]
```

```
?IN(BIGFX,BIGFY,BIGFXY)
DD=32 NUMD= 9 CARD= 33
[ 0.014, 0.04/ 0.121, 0.30]
```

```
?IN(BIGFZ,BIGFXY,BIGFXYZ)
DD=32 NUMD= 9 CARD= 12
[ 0.014, 0.02/ 0.116, 0.17]
```

```
?LIST(BIGFXYZ,1,15,'/2/')
5 QUAD 13 5890. 5890. 2490. -0. -0. -0.
6 QUAD 7 4610. 17200. 13000. -0. -0. -0.
11 QUAD 21 6090. 6090. 9280. -0. -0. -0.
12 QUAD 22 9020. 33700. 25200. -0. -0. -0.
17 QUAD 29 5800. 5800. 21300. -0. -0. -0.
18 QUAD 30 7300. 27300. 20600. -0. -0. -0.
23 QUAD 37 3370. 3370. 28000. -0. -0. -0.
23 QUAD 38 2040. 2040. 31100. -0. -0. -0.
30 QUAD 39 16400. 61200. 92900. -0. -0. -0.
36 QUAD 47 3190. 11900. 46500. -0. -0. -0.
41 QUAD 54 7960. 7960. 8920. -0. -0. -0.
42 QUAD 55 1590. 5940. 19000. -0. -0. -0.
```

Figure 3-7

```

?SYSTEM
#SSIGNOFF
#OFF AT 01:38.35    09-11-74
#ELAPSED TIME      18.657 MIN.          $5.91
#CPU TIME USED     19.294 SEC.          $1.74
#CPU STOR VMI      18.783 PAGE-MIN.     $1.02
#WAIT STOR VMI     23.974 PAGE-HR.
#APPROX. COST OF THIS RUN IS          $3.67
#DISK STORAGE      10423.191 PAGE-HR.   $3.44
#APPROX. REMAINING BALANCE:          $86.04
@

```

Figure 3-8

4.0 Dayfile Scenario

In this scenario, we consider a data base consisting of dayfile data from the computer complex at NASA Langley Research Center. The original data, taken from the Dayfile Summary Tape for 01 July 1974, is organized by jobs. Each entry in the data base represents a job and each job has over 50 variables. A data base, called DATA3, containing most of these variables is available; however, it is not used in this scenario because it is very cumbersome to use at an interactive terminal having only 70 or 80 columns. As a result a data base, called NEWDAY, with all of the jobs contained in DATA3 but with fewer variables (domains) for each job was created. The data base, NEWDAY, has domains as shown in figure 4-1. Only THRU, CPTIME, FL, OS CALLS, LINES, ESTIME, TAPE and CRT will be referenced in this scenario.

The first segment of the interactive session is shown in Figure 4-2. STDS is called with the RUN RBLG:STDS* command. We gain access to the previously created set NEWDAY (which is our entire data base) by using the GET (NEWDAY, 'NEWDAY') command. The cardinality of NEWDAY is given as 1243. We also use the GET command to gain access to two index sets, FPX and INX, which will be used in conjunction with the SUM command to generate statistical summaries. An SETFMT command is used to set up a format /1/ for use in the LIST command. Finally, the LIST command is used to print the first three elements of the data base as duplicated below with domains (column) headings.

NEWDAY DOMAINS

<u>DOMAIN</u>	<u>NAME</u>	<u>DESCRIPTION</u>
1 - 2	JOBNO	Job Identifier (7 characters)
3	THRU	Throughput - Total time (in minutes) that job is in the system (integer)
4	CPTIME	Total CPU time used in minutes (floating point)
5	CMTIME	Residence time (in minutes) in central memory not including ROLLOUT time (floating point)
6	PPTIME	PPU time used in minutes (floating point)
7	FL	Field length from job card (octal)
8	OSCALLS	Total number of O/S calls (floating point)
9	COST	Cost in dollars (integer)
10	LINES	Number of lines of output (integer)
11	ESTIME	Estimated time limit - job card (floating point)
12	CP	Machine: A, B, C, D, or Z
13	TAPE	Number of magnetic tape assignments (1 character)
14	CRT	Number of CRT assignments (1 character)
15	DC	Number of data cell control cards in job (1 character)

ORIGINAL PAGE IS
OF 1000 PAGES

Figure 4-1

In Figure 4-3 we call for statistical summaries of the entire data base with the SUM (NEWDAY, FPX, S1, 1) command and the SUM (NEWDAY, INX, S0, 0) command. In this version of STDS, it is necessary to ask for floating point summaries and integer summaries separately. Floating point domains are summarized with the FPX index set and a "1" in the last parameter position in the SUM command. Integer domains are summarized with INX index set and a "0" in the last parameter position of the SUM command. After the summaries, we calculate CP efficiency and average thruput for all jobs. From other sources we determined that the total production time for all five machines was approximately 117 hours for 01 July 1974. We can calculate CP efficiency (as a percentage) by taking the sum of CPTIME (domain 4) converting it to hours, dividing by total production time (117 hours) and multiplying by 100. The result is about 61% CP efficiency. The CALC command is an MTS command (not an STDS command) that allows the terminal user to evaluate expressions immediately. Average thruput for all jobs may be calculated by converting the average thruput (domain 3) to hours. The result is an average thruput of 4.1 hours. We use thruput as an approximation for turnaround since there are no figures available from which we can calculate turnaround. Turnaround is the time measured from the submission of a job to the time that the job is finished and is available for pickup by the user. Turnaround is equal to thruput plus the time necessary for manual handling of the job.

```
#RUN R3LG:STDS*
#EXECUTION BEGINS
```

```
** SET-THEORETIC INFORMATION MANAGEMENT INTERACTIVE INTERFACE **
[** STDS-1 ** VERS:UM02-17-74]
```

```
*** *** *** *** *** *** *** ***
*ENTER 4-CHARACTER UNIVERSE NAME *
:UNIV
*UNIVERSE UNIV HAS BEEN CREATED *
*** *** *** *** *** *** *** ***
```

```
"TAU-ON"
[COE-MAX(8-PAGE): 16 16 16]
```

```
?GET(NEWDAY,'NEWDAY')
DD=32 NUMD= 15 CARD= 1243
[ 0.004, 0.01/ 0.127, 0.29]
```

```
?GET(FPX,'DAY-FPINDEX')
DD=32 NUMD= 3 CARD= 5
[ 0.005, 0.00/ 0.133, 0.31]
```

```
?GET(INX,'DAY-INTINDEX')
DD=32 NUMD= 3 CARD= 4
[ 0.005, 0.01/ 0.133, 0.29]
```

```
?SETFMT(32,'(IX,2A4,14,F7.2,F6.1,F7.2,17,F5.1,14,16,F5.0,1X,4A1)')
FMT= 1
```

```
?LIST(NEWDAY,1,3,'/1/')
CT62591 154 41.40 140.4 40.51 152000 13236. 762 13782 123. A017
EXP0A01 34 0.01 0.4 0.35 52000 66. 5 0 1. D001
EXP0A02 735 1.06 4.2 2.80 145000 1576. 19 6835 5. A002
```

Figure 4-2

?SUM(NEWDAY,FPX,S1,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
4	0.425530E 04	3.42	0.02	142.71	9.41
5	0.191528E 05	15.43	0.07	357.08	30.01
6	0.725954E 04	5.84	0.07	351.64	13.39
8	0.271834E 07	2186.92	1.00	56985.00	4615.98
11	0.176702E 05	14.22	0.10	1296.30	67.01
DD=32 NUMD= 6 CARD= 5					
[1.505, 2.19/ 1.730, 38.15]					

?SUM(NEWDAY,INX,S0,0)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
3	0.308230E 06	247.97	1.00	1425.00	228.73
7	0.903568E 03	72692.50	7000.00	342000.00	37931.91
9	0.353600E 05	28.45	5.00	2336.00	126.67
10	0.410042E 07	3298.81	140.00	82551.00	6692.17
DD=32 NUMD= 6 CARD= 4					
[1.322, 2.15/ 1.538, 31.36]					

?SCOMMENT CALCULATE CP EFFICIENCY AND AVERAGE THRUPUT

?SCALC ((.42553E+4/60)/117)*100
 *=60.6163091168

?SCALC 247.97/60
 *=4.132833333333

Figure 4-3

In Figure 4-4 we have the interactive segment in which we isolate the jobs using one minute of CPTIME or less. We consider this to be a small demand for the CP resource. We isolate these jobs in a set called SMALLCP and notice that the cardinality of SMALLCP is 748. We may then perform the statistical summaries on SMALLCP. We may calculate the percentage of jobs by using the cardinalities of SMALLCP and NEWDAY. The result is that about 60% of the jobs are SMALLCP jobs. We then use the SUM of CPTIME (domain 4) for SMALLCP and NEWDAY to calculate percentage of CPTIME used. The result is that the SMALLCP jobs only use 4.3% of the CP resources. Finally, we calculate the average thrupt for SMALLCP jobs by converting mean THRU (domain 3) to hours. The result is 3.2 hours.

In the next segment of the session, Figure 4-5 we isolate jobs using five minutes or less of CPTIME. We can classify those as small and medium length jobs. We isolate these by using the RSLEA command in the set MEDCP. Note that SMALLCP is a subset of MEDCP. The cardinality of MEDCP is 1054. After asking for the floating point and integer summaries, we can calculate the job percentage, the CP percentage and the average thrupt in the same manner as before. The results are that about 85% of the job load are MEDCP jobs. They account for only 21% of the CP resources and their average turnaround is 3.7 hours.

In the Figure 4-6 segment, we isolate the jobs that do not require tape mounts. The command V\$(\$A, 32, ZERO) sets up a constant that is type character (as opposed to integer) and has a value of '0'. The domains TAPE, CRT, and DC were inadvertently input to STDS as type character rather than type integer; therefore it is necessary to retrieve it in character mode. The command RSEQ(13,NEWDAY,ZERO,NOTAPES) isolates the no-tape jobs in NOTAPES. The cardinality is 806. After the two standard

?%COMMENT ISOLATE JOBS WITH OPTIME .LE. 1.0 MINUTES

?RSLEA(4,NEWDAY,1.0,SMALLCP)

DD=32 NUMD= 15 CARD= 743

[0.140, 0.72/ 0.591, 4.711]

?SUM(SMALLCP,FPX,S1,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
4	0.183195E 03	0.24	0.02	0.98	0.26
5	0.338022E 04	4.52	0.07	116.23	8.04
6	0.191032E 04	2.55	0.07	54.15	4.31
8	0.566621E 06	757.51	1.00	17447.00	1679.99
11	0.451563E 04	6.04	0.10	75.00	10.49

DD=32 NUMD= 6 CARD= 5

[0.755, 1.92/ 0.971, 40.801]

?SUM(SMALLCP,INX,S0,0)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
3	0.143876E 06	192.35	1.00	1425.00	182.65
7	0.481597E 08	64384.68	7000.00	230000.00	32335.48
9	0.486100E 04	6.50	5.00	50.00	4.22
10	0.103142E 07	1445.75	140.00	20609.00	2211.48

DD=32 NUMD= 6 CARD= 4

[0.643, 5.39/ 0.852, 33.891]

?%COMMENT CALCULATE %JOBS LCP AND AV. THRUPUT

?%CALC 100*748/1243

#=60.17699115044

?%CALC 100*183.195/4255.3

#=4.305101872953

?%CALC 192.35/60

#=3.205833333333

Figure 4-4

?SCOMMENT ISOLATE JOBS WITH OPTIME .LE. 5.0 MINUTES

?RSLEA(4,NEWDAY,5.0,MEDCP)

DD=32 NUMD= 15 CARD= 1054

[0.173, 2.63/ 0.641, 9.331

?SUM(MEDCP,FPX,SI,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
4	0.901395E 03	0.86	0.02	4.86	1.14
5	0.858940E 04	8.15	0.07	357.08	18.39
6	0.449352E 04	4.26	0.07	351.64	12.35
8	0.149991E 07	1423.06	1.00	31043.00	2752.34
11	0.100075E 05	9.49	0.10	1296.30	58.86

DD=32 NUMD= 6 CARD= 5

[1.074, 6.63/ 1.293, 47.421

?SUM(MEDCP,INX,S0,0)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
3	0.234211E 06	222.21	1.00	1425.00	206.30
7	0.729292E 08	69192.75	7000.00	240000.00	35328.25
9	0.113230E 05	10.74	5.00	965.00	32.20
10	0.256065E 07	2429.46	140.00	27633.00	3894.53

DD=32 NUMD= 6 CARD= 4

[0.909, 2.58/ 1.107, 33.071

?SCOMMENT CALCULATE ZJOBS, 4CP, AND AV. THRUPT

?SCALC 100*1054/1243

#=84.79485116653

?SCALC 100*901.395/4255.0

#=21.18287782295

?SCALC 222.21/60

#=3.7035

Figure 4-5

?SCOMMENT ISOLATE NOTAPE JOBS

?V5(5A,32,ZERO)

:0

?RSEQ(13,NEWDAY,ZERO,NOTAPES)

DD=32 NUMD= 15 CARD= 306

[0.057, 0.097, 0.491, 1.401

?SUM(NOTAPES,FPX,S1,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
4	0.227963E 04	2.83	0.02	69.91	7.49
5	0.101005E 05	12.53	0.07	357.03	29.30
6	0.319857E 04	3.97	0.07	351.64	13.85
8	0.855233E 06	1061.08	1.00	35512.00	3047.36
11	0.109764E 05	13.62	0.10	1296.30	74.83

DD=32 NUMD= 6 CARD= 5

[0.811, 0.887, 1.019, 36.781

?SUM(NOTAPES,INX,S0,0)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
3	0.156442E 06	194.10	1.00	1150.00	172.34
7	0.577100E 08	71600.33	7000.00	250000.00	36971.63
9	0.169150E 05	20.99	5.00	1499.00	92.09
10	0.180654E 07	2241.36	140.00	26965.00	3376.33

DD=32 NUMD= 6 CARD= 4

[0.697, 0.717, 0.889, 29.511

?SCOMMENT CALCULATE ZJOBS, %CP, AND AV. THRUPUT

?SCALC 100*806/1243

#=64.84312148028

?SCALC 100*2279.63/4255.3

#=53.57154607195

?SCALC 194.1/60

#=3.235

Figure 4-6

summaries are generated, we calculate the job percentage, the CP percentage and the average thrupt. The results are that about 65% of the jobs are no-tape jobs. These require 54% of the CP time and have an average thrupt of 3.2 hours.

In the Figure 4-7 segment, we isolate the jobs requiring one or more tapes. This is easily done by "subtracting" the set NOTAPES from the set NEWDAY and forming the set TAPES. The relative complement command RL performs this function. After the summaries are requested, we again calculate the job and CP percentages and the average thrupt. The approximate results are that 35% of the jobs are tape jobs. They require 46% of the CP resources and the average thrupt is 5.8 hours.

At this point we would like to isolate the jobs that fit into our general concept of a small job. The criteria given here is somewhat arbitrary, but it serves to illustrate the power of a set-theoretic information system for isolating classes of jobs that the interrogator wishes to examine.

The criteria for a "small job" is as follows:

CPTIME	\leq	1.0 minute
FL	\leq	100000 words (octal)
LINES	\leq	2000 lines
OS CALLS	\leq	2000 O/S calls

and no CRT's assigned.

The Figure 4-8 segment demonstrates how this set called SMALLJOB, is isolated and Figure 4-9 and Figure 4-10 shows the analysis of small no-tape jobs and small tape jobs respectively. In Figure 4-11 we isolate the

?SCOMMENT ISOLATE TAPE JOBS

?RL(NEWDAY,NOTAPES,TAPES)

DD=32 NUMD= 15 CARD= 437

[0.372, 1.57/ 0.535, 1.81]

?SUM(TAPES,FPX,S1,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
4	0.197575E 04	4.52	0.23	142.71	12.11
5	0.908263E 04	20.73	0.13	246.52	30.57
6	0.406197E 04	9.30	0.13	94.59	11.73
8	0.186311E 07	4263.40	2.00	56985.00	6068.27
11	0.669441E 04	15.32	0.20	430.00	49.27

DD=32 NUMD= 6 CARD= 5

[0.443, 0.49/ 0.643, 36.29]

?SUM(TAPES,INX,S0,0)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
3	0.151788E 06	347.34	3.00	1425.00	280.71
7	0.326474E 08	74707.94	7000.00	342000.00	39562.93
9	0.184450E 05	42.21	5.00	2336.00	172.35
10	0.229389E 07	5249.17	52.00	32551.00	10024.86

DD=32 NUMD= 6 CARD= 4

[0.381, 0.40/ 0.572, 29.21]

?SCOMMENT CALCULATE ZJOBS, ZCP, AND AV. THRUPUT

?SCALC 100*437/1243

=35.15687851971

?SCALC 100*1975.75/4255.3

=46.4302339365

?SCALC 347.34/60

=5.789

Figure 4-7

?SCOMMENT ISOLATE SMALL JOBS

?RSLEA(7,NEWDAY,100000,SMALLFL)

DD=32 NUMD= 15 CARD= 993

[0.155, 0.41/ 0.610, 3.65]

?RSLEA(10,NEWDAY,2000,SMALLLINE)

DD=32 NUMD= 15 CARD= 770

[0.141, 0.35/ 0.578, 2.68]

?RSLEA(8,NEWDAY,2000.0,SMALLLOS)

DD=32 NUMD= 15 CARD= 951

[0.161, 0.21/ 0.602, 1.69]

?RSEQ(14,NEWDAY,ZERO,NOCRT)

DD=32 NUMD= 15 CARD= 1227

[0.077, 0.54/ 0.536, 2.12]

?IN(SMALLCP,SMALLFL,SMALLJOB)

DD=32 NUMD= 15 CARD= 658

[0.121, 1.38/ 0.262, 1.90]

?IN(SMALLJOB,SMALLLINE,SMALLJOB)

DD=32 NUMD= 15 CARD= 522

[0.099, 0.94/ 0.241, 1.15]

?IN(SMALLJOB,SMALLLOS,SMALLJOB)

DD=32 NUMD= 15 CARD= 495

[0.098, 0.48/ 0.241, 0.67]

?IN(SMALLJOB,NOCRT,SMALLJOB)

DD=32 NUMD= 15 CARD= 494

[0.106, 0.57/ 0.246, 0.76]

Figure 4-8

small field length jobs in SMALLFL (card = 993), the small output jobs in SMALLINE (card = 770), the small O/S call jobs in SMALLLOS (card = 951), and the NO-CRT jobs in NOCRT (card = 1227). We may now form the intersection

$$\text{SMALLJOB} = \text{SMALLCP} \cap \text{SMALLFL} \cap \text{SMALLINE} \cap \text{SMALLLOS} \cap \text{NOCRT}$$

by performing four successive IN functions as shown. The (final) cardinality of SMALLJOB is 494 or about 39.7% of the jobs.

In the Figure 4-9 segment, we isolate the small no-tape jobs in the set NOTAPESM. This is the class of jobs that one would expect to have the shortest turnaround. We perform the standard statistical summaries as shown. The HIGH figure for THRU (domain 3) indicates that turnaround can go over 12 hours for even the shortest jobs. We calculate the job percentage, the CP percentage and the average thruput. The results are that the small no-tape jobs account for about 30% of the jobs and use only 1.3% of the CP resources. The average thruput is 2.1 hours. Note that the total CP time necessary to run these 376 jobs was less than one hour. One 6000 machine devoted to small jobs of this class during normal working hours would run between 1500 and 3000 jobs in an eight hour period depending on the CP power and efficiency.

In the Figure 4-10 segment, we isolate the small, tape jobs by intersecting SMALLJOB with TAPES forming SMALTAPE (card = 118). The statistical summaries are requested. We calculate the job percentage, the CP percentage, and the average thruput. The results are that almost 9.5% of the jobs are small tape jobs, they use almost 0.6% of the CP resources and the average thruput is 4.4 hours. In addition, the HIGH value of THRU (domain 3) indicates that turnaround for a small tape job can approach 24 hours.

?SCOMMENT ISOLATE SMALL JOBS WITHOUT TAPES

?IN(SMALLJOB,NOTAPES,NOTAPESM)

DD=32 NUMD= 15 CARD= 376

I 0.034, 0.567 0.212, 0.711

?SUM(NOTAPESM,FPX,S1,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
4	0.561680E 02	0.15	0.02	0.93	0.20
5	0.100656E 04	2.68	0.07	109.55	6.56
6	0.498717E 03	1.33	0.07	11.68	1.69
8	0.879460E 05	233.90	1.00	1841.00	309.99
11	0.193047E 04	5.13	0.10	62.90	9.68

DD=32 NUMD= 6 CARD= 5

I 0.382, 0.427 0.590, 36.301

?SUM(NOTAPESM,INX,S0,0)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
3	0.474290E 05	126.14	1.00	765.00	96.19
7	0.200489E 08	53321.53	7000.00	100000.00	14192.30
9	0.192100E 04	5.11	5.00	9.00	0.45
10	0.183984E 06	439.32	140.00	1978.00	571.30

DD=32 NUMD= 6 CARD= 4

I 0.328, 0.337 0.521, 29.221

?SCOMMENT CALCULATE ZJOB, ZCP, AND AV. THRUPUT

?SCALC 100*376/1243

=30.24939662107

?SCALC 100*56.168/4255.3

=1.319953939792

?SCALC 126.14/60

=2.102333333333

Figure 4-9

?SCOMMENT ISOLATE SMALL JOBS WITH TAPES

?INC(SMALLJOB,TAPES,SMALTAPE)

DD=32 NUMD= 15 CARD= 118

[0.056, 0.64/ 0.176, 0.79]

?SUM(SMALTAPE,FPX,S1,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
4	0.251997E 02	0.21	0.23	0.97	0.23
5	0.611853E 03	5.19	0.13	39.50	6.03
6	0.272228E 03	2.31	0.13	12.63	2.20
8	0.609370E 05	516.42	2.00	1913.00	524.16
11	0.839195E 03	7.11	0.20	50.00	12.35

DD=32 NUMD= 6 CARD= 5

[0.124, 0.15/ 0.330, 36.03]

?SUM(SMALTAPE,INX,S0,0)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
3	0.313710E 05	265.86	3.00	1425.00	229.13
7	0.640648E 07	54292.18	7000.00	100000.00	18329.57
9	0.647000E 03	5.43	5.00	29.00	2.52
10	0.671560E 05	569.12	52.00	1964.00	548.31

DD=32 NUMD= 6 CARD= 4

[0.108, 0.11/ 0.299, 29.03]

?SCOMMENT CALCULATE %JOBS, %CP, AV. THRUPUT

?SCALC 100*118/1243

#=9.493161705551

?SCALC 100*25.1997/4255.3

#=.5921956148803

?SCALC 265.86/60

#=4.431

Figure 4-10

Figure 4-12 gives a manually created summary tabular form of the results obtained during the session. Figure 4-11 provides some information about the length and cost of the session. The total wall clock time for the session (as given by elapsed time) was less than an hour and the cost of the session (at university rates) was just over \$10.00.

?SCOMMENT END OF SESSION

?SYSTEM

#UN

#SSIGNOFF

#OFF AT 03:51.58 09-13-74

#ELAPSED TIME 51.653 MIN. \$2.52

#CPU TIME USED 38.775 SEC. \$3.50

#CPU STOR VMI 75.384 PAGE-MIN. \$4.03

#WAIT STOR VMI 115.819 PAGE-HR.

#DRUM READS 1590

#APPROX. COST OF THIS RUN IS \$10.11

#DISK STORAGE 3244.564 PAGE-HR. \$1.07

#APPROX. REMAINING BALANCE: \$51.50

>

Figure 4-11

<u>JOB CLASS</u>	<u>% JOBS</u>	<u>% CP</u>	<u>AVG. THRUPUT</u> (Hours)
NEWDAY	100.0%	100.0%	4.13
SMALLCP	60.18%	4.3%	3.21
MEDCP	84.79%	21.18%	3.70
NOTAPES	64.84%	53.57%	3.24
TAPES	35.16%	46.43%	5.79
NOTAPESM	30.25%	1.32%	2.10
SMALTAPE	9.49%	0.59%	4.43

CP Efficiency = 61%

Figure 4-12

5.0 VGH Scenario

The first segment of the VGH interactive session is shown in Figure 5-1. We are in a position to retrieve the data base from mass storage and make it accessible to STDS. The `GET` command performs this function. Then, we list the first five elements in the data base. Each column in the listing represents one domain of the set. Domain number one is the aircraft type, domain two is the serial number, and so on as shown in Figure 5-2. Only the serial number, the gust indicator, the indicated air-speed, the altitude, and the acceleration are used in this scenario.

For this scenario, we have prepared an appropriate index set in advance and stored it as a permanent file. We can retrieve it with the command: `GET(INDEX,'VGH-INDEX')`. The set, `INDEX`, provides for statistical summaries to be generated for the domains, `IAS (8)`, `Q(9)`, `ALT(10)`, `P(11)`, and `ACC(12)`. Now, we may generate a statistical summary of the entire data base (i. e. the set `VGH`) with the `SUM` command.

#RUN RBLG:STDS#
#EXECUTION BEGINS

** SET-THEORETIC INFORMATION MANAGEMENT INTERACTIVE INTERFACE **
[** STDS-I ** VERS:UM02-17-74]

*** **
*ENTER 4-CHARACTER UNIVERSE NAME *
:UNIV
*UNIVERSE UNIV HAS BEEN CREATED *
*** **

"TAU-ON"
[CORE-MAX(8-PAGE): 16 16 16]

?SETFMT(32,'(214,15,13,12,1X,215,5F8.2)')
FMT= 1

?GET(VGH,'HBPT:VGH-SET')
DD=32 NUMD= 12 CARD= 7263
[0.005, 0.01/ 0.321, 0.751

?LIST(VGH,1,5,'/1/')
128 79 1 1 0 10401 1 73.90 18.56 8.60 2115.57 0.0
128 79 1 1 0 10402 1 106.67 38.85 2190.53 1954.14 0.0
128 79 2 1 0 10402 2 80.45 22.00 530.42 2075.98 0.0
128 79 2 1 0 10402 2 112.24 43.02 2006.59 1967.21 0.0
128 79 3 1 0 10402 3 94.02 30.09 844.90 2052.45 0.0

?GET(INDEX,'VGH-INDEX')
DD=32 NUMD= 3 CARD= 5
[0.005, 0.01/ 0.144, 0.351

?SUM(VGH,INDEX,S1,1)
*** KEY SUM MEAN LOW HIGH STD ***
8 0.660393E 06 90.99 -191.89 126.53 12.53
9 0.263569E 06 26.72 -3.24 54.78 7.47
10 0.133710E 08 1840.97 -214.89 8517.24 1405.49
11 0.143852E 08 1980.61 11.39 2132.77 100.96
12 0.381371E 03 0.05 -1.53 1.89 0.35
DD=32 NUMD= 6 CARD= 5
[8.558, 12.48/ 8.827, 48.44]

Figure 5-1

ORIGINAL PAGE IS
OF POOR QUALITY

VGH DOMAINS

1. T - Aircraft Type Identifier
2. S - Aircraft Serial Number
3. K - A counter used to verify that data was
input correctly into the information system.
4. F - Flight Number
5. G - Gust Indicator; 0 is no data, 1 is calm, 2 is rough.
6. C1 - Clock - Upper 5 digits
7. C2 - Clock - Lower 5 digits
8. IAS - Indicated Airspeed (knots)
9. Q - Impact Pressure(PSF)
10. ALT - Pressure Altitude(Feet)
11. P - Static Pressure (PSF)
12. ACC - Acceleration (g's)

Figure 5-2

In Figure 5-1, we have a summary of the entire data base, VGH, and we notice that we have some bad data points. Under KEY=8 (IAS), the LOW figure is -191.89 and under KEY=10, the LOW figure is -214.89. Both of these fields should contain only non-negative numbers. Although it is possible to make corrections, it is somewhat cumbersome in STDS. As a result, we take the general attitude that bad data points will be deleted. In Figure 5-3, we isolate the negative IAS with the first RSLTA command. Since there is only one, we list the set for the purpose of examining it. Next, we isolate the negative altitudes. Since there are thirty, we only look at the first five. Next, with the relative complement command, RL, we form a new set VGH2 omitting the bad IAS data point. And again using the RL command, we form VGH3 omitting all the bad altitude data points. The PUT command allows us to save our updated data base as an MTS permanent file. Finally, we perform a statistical summary on the new data base in order to verify our deletions.

Figure 5-4 describes housekeeping operations that may be performed from time to time. The LISTU command lists the sets that are currently active. Under the current version of STDS, the user may have only twenty sets active concurrently. As a result, it is necessary now and then to free sets that we are no longer using or save them as permanent files with the PUT command. The final GET command retrieves our updated version of the data base with our preferred name, VGH.

Figure 5-5 describes the procedure for isolating items by serial number. Currently, the VGH Summary Program is used to obtain statistical information from the VGH data tapes. Each tape contains some flight

?SCOMMENT ERROR DETECTION AND CORRECTION

?RSLTA(8,VGH,0.0,IASNEG)

DD=32 NUMD= 12 CARD= 1

[0.320, 2.37/ 1.726, 6.88]

?LIST(IASNEG,1,1,'/1/')

128 79 952 4 1 970258196 -191.89 -3.24 3625.57 1853.51 -0.44

?RSLTA(10,VGH,0.0,ALTNEG)

DD=32 NUMD= 12 CARD= 30

[0.320, 0.32/ 1.707, 4.35]

?LIST(ALTNEG,1,5,'/1/')

128	79	43	3	0	10402	28	79.72	21.60	-186.11	2130.55	0.0
128	79	119	5	0	10002	46	76.14	19.71	-164.51	2128.89	0.0
128	79	123	5	0	10002	50	72.28	17.76	-214.89	2132.77	0.0
128	79	126	5	0	10002	53	62.43	13.27	-157.31	2123.34	0.0
128	79	129	5	0	10002	56	68.73	16.05	-99.63	2123.90	0.0

?RL(VGH,IASNEG,VGH2)

DD=32 NUMD= 12 CARD= 7262

[1.494, 6.76/ 1.869, 9.96]

?RL(VGH2,ALTNEG,VGH3)

DD=32 NUMD= 12 CARD= 7232

[0.360, 2.69/ 0.680, 5.66]

?PUT(VGH3,'NEWVGH')

% DD=32 NUMD= 12 CARD= 7232

[1.590, 9.70/ 1.965, 11.42]

?SUM(VGH3,INDEX,S2,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
8	0.659120E 06	91.14	30.43	126.53	11.92
9	0.203090E 06	23.77	3.14	54.73	7.41
10	0.133703E 08	1843.84	1.28	3517.24	1402.67
11	0.143196E 03	1930.03	11.39	2116.12	100.74
12	0.377032E 03	0.05	-1.53	1.89	0.35

DD=32 NUMD= 6 CARD= 5

[7.323, 8.63/ 7.524, 44.52]

Figure 5-3

?SCOMMENT HOUSEKEEPING OPERATIONS

?LISTU

NAME	TYPE	CRE#	CARD	DD	NOO	STATUS	SIZE	LOC
VGH	TAU	1	7263	32	12	1	85.11	HBPT:VGH-SET
INDEX	TAU	2	5	32	3	1	0.01	VGH-INDEX
S1	TAU	3	5	32	6	2	0.03	CORE
IASNEG	TAU	4	1	32	12	2	0.01	CORE
ALTNEG	TAU	5	30	32	12	2	0.35	CORE
VGH2	TAU	6	7262	32	12	2	85.10	CORE
VGH3	TAU	7	7232	32	12	2	84.75	CORE
S2	TAU	8	5	32	6	2	0.03	CORE

?FREE(VGH)

?FREE(S1)

?FREE(IASNEG)

?FREE(ALTNEG)

"FREE" IS NOT DEFINED.

?FREE(ALTNEG)

?FREE(VGH2)

?FREE(VGH3)

?FREE(S2)

?GET(VGH, 'NEWVGH')

DD=32 NUMO= 12 CARD= 7232

[0.005, 0.007, 0.062, 0.15]

Figure 5-4

ORIGINAL PAGE IS
OF POOR QUALITY

?SCOMMENT ISOLATE BY SERIAL NUMBER

?RSEQ(2,VGH,185,SER185)

DD=32 NUMD= 12 CARD= 2500

[0.218, 1.37/ 1.719, 7.571

?LIST(SER185,1,5,'/1/')

150	185	1	7	2	892516353	78.64	21.03	1016.69	2039.63	0.85
150	185	2	7	2	892516354	79.01	21.22	1016.69	2039.63	0.79
150	185	3	7	2	892516355	79.91	21.70	1032.76	2038.45	1.12
150	185	4	7	2	892516356	68.31	15.86	1024.72	2039.04	0.59
150	185	5	7	2	892516357	70.44	16.85	961.34	2043.74	0.76

?SUM(SER185,INDEX,S1,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
8	0.212446E 06	84.98	60.60	114.67	6.16
9	0.617426E 05	24.70	12.47	44.94	3.34
10	0.292511E 07	1170.04	552.38	2299.10	319.59
11	0.507010E 07	2028.04	1946.43	2074.34	30.17
12	0.273578E 03	0.11	-1.53	1.89	0.51

DD=32 NUMD= 6 CARD= 5

[2.550, 2.60/ 2.751, 38.421

?RSEQ(2,VGH,79,SER79)

DD=32 NUMD= 12 CARD= 2623

[0.227, 1.66/ 1.739, 6.531

?LIST(SER79,1,5,'/1/')

128	79	1	1	0	10401	1	73.90	18.56	8.60	2115.57	0.0
128	79	1	1	0	10402	1	106.67	38.85	2190.53	1954.14	0.0
128	79	2	1	0	10402	2	80.45	22.00	530.42	2075.98	0.0
128	79	2	1	0	10402	2	112.24	43.02	2206.59	1967.21	0.0
128	79	3	1	0	10402	3	94.02	30.09	844.90	2052.45	0.0

?SUM(SER79,INDEX,S2,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
8	0.252574E 06	96.11	34.05	126.53	13.63
9	0.844214E 05	32.12	3.94	54.73	3.44
10	0.490538E 07	1866.58	1.28	8517.24	1559.04
11	0.524037E 07	1973.83	11.39	2116.12	116.52
12	0.109953E 03	0.04	-0.82	1.02	0.21

DD=32 NUMD= 6 CARD= 5

[2.665, 2.72/ 2.865, 38.601

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 5-5

data for a particular serial number. The VGH Summary Program passes the tape and generates the summary. As a result all summaries are by serial number and it is difficult to obtain answers to queries that would require access to data for many serial numbers at one time. Appendix A gives some information that is included in the output record for the VGH Summary Program and we propose to answer a number of those types of queries using STDS; but in fact there is a large class of queries that would be useful that are not included in Appendix A because they cannot easily be obtained from records sorted by serial number. An example of one of these is: Give me all occurrences for all aircraft where the acceleration is greater than or equal to 1 g. This query is considered later in this section.

However, in many cases it will be desirable to isolate data by serial number. First, we isolate the data for serial number 185 with the command: RSEQ(2,VGH,185,SER185). Then, we list the first five elements and obtain a statistical summary. From this, we can directly obtain the average IAS (84.98) and the average altitude (1170.04) as well as the high and low values. We perform the same operations for serial number 79; an RSEQ followed by a LIST followed by a SUM.

Figures 5-6 and 5-7 describe the procedure for isolating items by intervals in some domain. A number of items in Appendix A depend upon retrieval by interval; time interval, altitude interval and velocity interval. In this section, we demonstrate isolating readings by altitude intervals. The same technique may be applied for time and velocity intervals.

?SCOMMENT INTERVAL ISOLATION - ALTITUDE

?BRSA(10,VEH,2000.0,3000.0,INTERVAL)

DD=32 NUMD= 12 CARD= 1240

[0.230, 1.82/ 1.679, 6.85]

?LIST(INTERVAL,1,5,'/1/')

128	79	1	1	0	10402	1	106.67	38.85	2190.53	1954.14	0.0
128	79	2	1	0	10402	2	112.24	43.02	2006.59	1967.21	0.0
128	79	3	1	0	10402	3	98.34	32.95	2022.57	1966.08	0.0
128	79	4	1	0	10402	4	106.51	38.73	2118.52	1959.26	0.0
128	79	5	1	0	10402	5	103.17	36.33	2134.51	1958.12	0.0

?SUM(INTERVAL,INDEX,S3,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
8	0.119562E 06	96.42	34.81	121.58	13.80
9	0.401008E 05	32.34	4.11	50.52	8.32
10	0.302504E 07	2439.55	2002.65	2994.96	293.33
11	0.239890E 07	1934.59	11.39	1967.49	65.59
12	0.320992E 01	0.00	-1.10	1.44	0.23

DD=32 NUMD= 6 CARD= 5

[1.266, 1.32/ 1.467, 37.20]

?IN(INTERVAL,SER79,INT79)

DD=32 NUMD= 12 CARD= 514

[0.161, 0.23/ 0.294, 0.33]

?LIST(INT79,1,5,'/1/')

128	79	1	1	0	10402	1	106.67	38.85	2190.53	1954.14	0.0
128	79	2	1	0	10402	2	112.24	43.02	2006.59	1967.21	0.0
128	79	3	1	0	10402	3	98.34	32.95	2022.57	1966.08	0.0
128	79	4	1	0	10402	4	106.51	38.73	2118.52	1959.26	0.0
128	79	5	1	0	10402	5	103.17	36.33	2134.51	1958.12	0.0

?SUM(INT79,INDEX,S4,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
8	0.513190E 05	99.34	34.81	121.58	14.43
9	0.178380E 05	34.70	4.11	50.52	8.97
10	0.124567E 07	2425.44	2006.59	2993.55	273.29
11	0.993926E 06	1933.71	11.39	1967.21	87.12
12	0.592000E 01	0.01	-0.71	1.00	0.16

DD=32 NUMD= 6 CARD= 5

[0.527, 0.53/ 0.728, 35.29]

Figure 5-6

?IN(INTERVAL,SER185,INT185)

DD=32 NUMD= 12 CARD= 71

[0.120, 0.63/ 0.239, 0.771

?LIST(INT185,1,5,'/1/')

150	185	1087	3	1	970258350	80.76	22.18	2126.21	1958.71	0.40
150	185	1088	3	1	970258351	79.91	21.70	2159.15	1956.37	-0.54
150	185	1089	3	1	970258352	85.41	24.81	2299.10	1946.43	0.40
150	185	1090	3	1	970258353	85.73	25.00	2249.71	1949.94	-0.41
150	185	1091	3	1	970258354	79.10	21.27	2126.21	1958.71	-0.56

?SUM(INT185,INDEX,SS,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
8	0.604273E 04	85.11	63.25	94.89	5.45
9	0.175677E 04	24.74	13.60	38.64	3.04
10	0.151823E 06	2138.36	2002.65	2299.10	79.88
11	0.139006E 06	1957.83	1946.43	1967.49	9.32
12	-.201997E 01	-0.03	-0.76	0.63	0.47

DD=32 NUMD= 6 CARD= 5

[0.079, 0.08/ 0.279, 36.031

Figure 5-7

First, we isolate all of the readings in the data base that have an altitude value between 2000.0 and 3000.0 feet. This is done with the BRSA command forming the set INTERVAL. We then list the first five items and perform a summary. In order to isolate the readings associated with serial number 79, we intersect the sets, INTERVAL and SER79 forming the new set INT79. This set contains all the readings for serial 79 in the altitude range 2000 to 3000 feet. From the cardinality indicator, we immediately know that there are 514 readings in the set and a SUM command provides us with some additional information; the high, low and average IAS (121.58, 34.81, 99.84), and the high and low accelerations (-0.71, 1.00). One immediate advantage of an interactive system over a batch environment is the following. If the interrogator spots an area that needs further investigation, he has at his disposal the capability to immediately search to the root of the problem.

Next, we isolate the readings for serial 185 by intersecting the set of interval readings, INTERVAL, with the set of readings for serial number 185, SER185. As before, this isolates the readings for serial 185 in the altitude interval. A LIST and a SUM provide a sample of the set contents and the usual statistical information.

Figures 5-8 and 5-9 describe the procedure for isolating maneuver accelerations that exceed given thresholds. We will consider only the positive accelerations. However, we would normally be interested in the

?SCOMMENT ISOLATE ACCELERATIONS

?RSGTA(12,VGH,0.0,POSAC)

DD=32 NUMD= 12 CARD= 1988

[0.512, 0.92/ 2.000, 6.47]

?IN(POSAC,SER185,POSAC185)

DD=32 NUMD= 12 CARD= 1463

[0.246, 1.21/ 0.406, 1.56]

?SUM(POSAC185,INDEX,S1,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
8	0.122841E 06	83.97	62.92	114.67	6.69
9	0.353081E 05	24.13	13.45	44.94	3.64
10	0.167538E 07	1145.17	576.00	2299.10	309.32
11	0.296981E 07	2029.94	1946.43	2072.57	34.60
12	0.767443E 03	0.52	0.39	1.89	0.15

DD=32 NUMD= 6 CARD= 5

[1.484, 1.55/ 1.685, 37.38]

?IN(POSAC,SER79,POSAC79)

DD=32 NUMD= 12 CARD= 354

[0.151, 1.18/ 0.284, 1.37]

?SUM(POSAC79,INDEX,S2,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
8	0.330032E 05	93.23	44.93	119.25	12.33
9	0.106664E 05	30.13	6.84	48.57	7.45
10	0.398555E 06	1125.86	1.28	3375.88	816.17
11	0.719444E 06	2032.33	1836.25	2116.12	59.59
12	0.171048E 03	0.48	0.40	1.02	0.11

DD=32 NUMD= 6 CARD= 5

[0.364, 0.44/ 0.566, 36.25]

Figure 5-8

?SCOMMENT ISOLATE 1G ACCELERATIONS (AND OVER)

?RSGEA(12,POSAC,1.0,ONEG)

DD=32 NUMD= 12 CARD= 34

[0.091, 0.10/ 0.198, 0.24]

?LIST(ONEG,1.5,'/1/')

128	79	1079	12	2	852516353	112.17	42.96	1402.94	2011.37	1.02
128	79	1154	21	2	852516356	85.15	24.65	2864.72	1906.25	1.30
128	288	2089	40	2	852516356	72.44	17.84	2117.13	1959.36	1.44
150	185	3	7	2	892516355	79.91	21.70	1032.76	2038.45	1.12
150	185	7	7	2	892516359	74.47	18.84	937.76	2045.51	1.17

?SUM(ONEG,INDEX,S3,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
8	0.260615E 04	76.65	64.58	114.67	12.11
9	0.696488E 03	20.48	14.16	44.94	7.34
10	0.376145E 05	1106.31	764.82	2864.72	380.95
11	0.691284E 05	2033.19	1906.25	2058.45	27.73
12	0.383193E 02	1.13	1.00	1.89	0.17

DD=32 NUMD= 6 CARD= 5

[0.041, 0.07/ 0.243, 36.07]

?RSGTA(12,ONEG,1.5,ONESG)

DD=32 NUMD= 12 CARD= 1

[0.004, 0.00/ 0.089, 0.09]

?LIST(ONESG,1.1,'/1/')

150	185	30	7	2	892516383	100.34	34.29	1024.72	2039.04	1.39
-----	-----	----	---	---	-----------	--------	-------	---------	---------	------

Figure 5-9

magnitude of the acceleration in both positive and negative directions. The positive accelerations may be isolated using the RSGTA command. To isolate accelerations for serial 185 we intersect the positive acceleration set, POSAC, with the set of readings for serial 185, SER185. Statistics for this set may be obtained with the SUM command. Likewise, the accelerations for serial 79 may be obtained by intersecting the sets, POSAC and SER 79 and statistics may be obtained using the SUM command. may be obtained using the SUM command.

As mentioned previously, it might be desirable to isolate all occurrences for all aircraft of accelerations equal to or over 1 g. This may be accomplished by using an RSGEA command forming a new set, ONEG. We know immediately from the cardinality indicator that there are 34 occurrences. We may want to list all of them out at the terminal or only a few (as is shown) and perform a summary. The summary indicates that the high is 1.89; so we might decide to look at all over 1.5 g. This is accomplished using an RSGTA command and we find that the 1.89 reading is the only one and we look at it using the LIST command. It occurred on serial 185.

By now the reader will be familiar with the routine of (1) isolating a set describing a condition, (2) intersecting it with a set of readings for a particular serial number, and (3) obtaining statistics for the derived set. In this section, we follow an identical strategy in examining the rough air readings as shown in Figure 5-10.

Several items in Appendix A call for statistical analysis of sets that we have previously derived under Rough Air, Intervals, and

Accelerations. Figure 5-11 demonstrates the procedure for combining the rough air and altitude interval sets to determine the occurrences of rough air encounters in the altitude interval 2000 to 3000 feet for serials 185 and 79.

The time and cost of this session is shown in Figure 5-12. Note that the connect time from initial dial-up to signoff was one hour and fifteen minutes and the central processor time was about 70 seconds on an IBM 360/67. The dollar values are for a university environment and would be somewhat higher in a commercial environment.

? COMMENT ROUGH AIR

?RSEQ(5,VGH,2,ROUGH)

DD=32 NUMD= 12 CARD= 493

[0.164, 0.58/ 1.596, 4.93]

?IN(ROUGH,SER79,RUF79)

DD=32 NUMD= 12 CARD= 146

[0.133, 0.91/ 0.254, 1.11]

?LIST(RUF79,1,5,'/1/')

128	79	1064	3 2	892516353	87.18	25.86	1967.61	1970.05	0.59
128	79	1065	5 2	852516353	105.77	38.21	650.10	2067.03	0.53
128	79	1066	5 2	852516354	96.95	32.01	807.41	2055.26	0.43
128	79	1067	5 2	852516355	92.92	29.40	717.43	2061.99	0.67
128	79	1068	5 2	852516356	98.69	33.18	754.94	2059.19	0.59

?SUM(RUF79,INDEX,S1,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
8	0.126607E 05	86.72	34.05	119.25	20.21
9	0.394350E 04	27.01	3.94	48.57	10.35
10	0.240504E 06	1647.29	8.60	3875.88	911.22
11	0.291163E 06	1994.27	1836.25	2115.57	66.10
12	0.566793E 02	0.39	-0.72	1.02	0.40

DD=32 NUMD= 6 CARD= 5

[0.155, 0.18/ 0.356, 36.03]

?IN(ROUGH,SER185,RUF185)

DD=32 NUMD= 12 CARD= 301

[0.052, 0.14/ 0.175, 0.28]

?LIST(RUF185,1,5,'/1/')

150	185	1	7 2	892516353	78.64	21.03	1016.69	2039.63	0.85
150	185	2	7 2	892516354	79.01	21.22	1016.69	2039.63	0.79
150	185	3	7 2	892516355	79.91	21.70	1032.76	2038.45	1.12
150	185	4	7 2	892516356	68.31	15.86	1024.72	2039.04	0.59
150	185	5	7 2	892516357	70.44	16.85	961.34	2043.74	0.76

?SUM(RUF185,INDEX,S2,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
8	0.226744E 05	75.33	62.92	114.67	6.47
9	0.584959E 04	19.43	13.45	44.94	3.52
10	0.287970E 06	956.71	583.87	1313.68	120.84
11	0.615275E 06	2044.10	2017.90	2071.98	8.50
12	0.214323E 03	0.71	0.40	1.89	0.21

DD=32 NUMD= 6 CARD= 5

[0.313, 0.63/ 0.516, 36.55]

Figure 5-10

?SCOMMENT COMBINATIONS - ROUGH AIR IN THE INTERVAL 2-3 THOUSAND FT.

?IN(RUF185,INT185,IR185)

NULL SET!

[0.022, 0.05/ 0.118, 0.18]

?IN(RUF79,INT79,IR79)

DD=32 NUMD= 12 CARD= 21

[0.038, 0.20/ 0.149, 0.31]

?LIST(IR79,1,5,'/1/')

128	79	1077	9	2	852516353	44.15	6.62	2086.53	1961.53	-0.53
128	79	1078	9	2	852516354	42.02	6.00	2543.22	1929.09	-0.55
128	79	1118	12	2	852516392	76.90	20.11	2222.55	1951.87	0.79
128	79	1119	12	2	852516393	40.47	5.56	2535.17	1929.66	-0.55
128	79	1120	12	2	852516394	58.50	11.63	2495.06	1932.51	0.45

?SUM(IR79,INDEX,S3,1)

*** KEY	SUM	MEAN	LOW	HIGH	STD ***
8	0.148883E 04	70.90	34.81	103.09	22.65
9	0.395909E 03	18.85	4.11	36.27	10.90
10	0.545536E 05	2597.79	2086.53	2985.50	264.94
11	0.404294E 05	1925.21	1897.67	1961.53	18.35
12	0.560000E 01	0.27	-0.62	1.00	0.52

DD=32 NUMD= 6 CARD= 5

[0.028, 0.03/ 0.229, 36.03]

Figure 5-11

?SCOMMENT END OF STDS SESSION

?SYSTEM

#UN

#SIG

#OFF AT 01:54.49 09-04-74

#ELAPSED TIME 74.383 MIN. \$3.63

#CPU TIME USED 69.983 SEC. \$6.32

#CPU STOR VMI 239.938 PAGE-MIN. \$13.00

#WAIT STOR VMI 199.521 PAGE-HR.

#DRUM READS 866

#APPROX. COST OF THIS RUN IS \$22.95

#DISK STORAGE 3452.935 PAGE-HR. \$1.14

#APPROX. REMAINING BALANCE: \$131.80

Figure 5-12

5.1 USING STDS FROM FORTRAN

The reader will notice in Appendix A that there is a class of queries that are not directly answerable from an interactive system. For example, true airspeed does not appear explicitly in the data base; it must be calculated from impact pressure (Q) pressure altitude (ALT) and static pressure (P). The following equations in FORTRAN are used to obtain TAS.

```
AARG = 518.688 - .00356617 *ALT
```

```
SS    = 29.04425 * SQRT (AARG)
```

```
XM    = SQRT(5*(Q/PS + 1.0)**.2857-1))
```

```
TAS   = SS*XM
```

In this subsection, we demonstrate how one may call STDS from an executing (FORTRAN) program. This provides the user with the capability to perform calculations based on items in his data base. Specifically, we demonstrate here a program that calculates average TAS in the altitude interval 8000.0-9000.0 feet. The interaction described herein was a session in itself from signon to signoff. The time and costs are given in Figure 5 - 15.

The program was keyed in and debugged in previous sessions not detailed in this report. Figure 5 - 13 begins immediately after sign-on as we call the MTS text editor and request it to access the file TAS in which our FORTRAN code is stored. The command P *F *L is an editor command that requests that the text be printed from the first line (*F) to the last line (*L). The editor prints the entire program and finally is asked to terminate execution with the command, STOP. Now, we may consider the logic of the FORTRAN program itself. We assume the reader is familiar

#ED TAS

:P *F *L

```

:      1      C** PROGRAM TO CALCULATE TAS
:      2      INTEGER CARD
:      2.25    CALL UNIV('QQQQ ')
:      3      CALL GET(VGH, NEWVGH ')
:      3.25    IC=CARD(VGH)
:      3.5     PRINT 94,IC
:      3.6     94 FORMAT(1X,'CARD OF NEWVGH = ',I6)
:      4      CALL BRSA(10,VGH,8000.0,9000.0,INT)
:      4.1     IC = CARD(INT)
:      4.25    PRINT 95, IC
:      4.5     95 FORMAT(' CARD OF INT = ', I6)
:      5      TSUM = 0.0
:      6      N = 0
:      7      100 CONTINUE
:      7.25    K = CARD(INT)
:      8      IF( K .EQ. 0) GO TO 500
:      9      N = N + 1
:     10      C** GET Q PS AND ALT
:     11      CALL GETELM(1,INT,33,4,Q)
:     12      CALL GETELM(1,INT,37,4,ALT)
:     13      CALL GETELM(1,INT,41,4,PS)
:     14      C** CALCULATE TAS
:     14.25    AARG = 518.688-.00356617*ALT
:     15      88 = 29.04425*SQRT(AARG)
:     16      XM = SQRT(5*((Q/PS + 1.0)** .2857 -1))
:     17      TAS = SS*XM
:     18      TSUM = TSUM + TAS
:     19      C** DELETE THE ELEMENT FROM THE SET
:     20      CALL SUBSET(INT,1,1,1,UNIT)
:     21      CALL RL(INT,UNIT,INT)
:     22      GO TO 100
:     23      500 CONTINUE
:     24      C** FINISHED WITH SET
:     25      TMEAN = TSUM/N
:     26      PRINT 90, TMEAN
:     27      90 FORMAT(1X,'AVERAGE TAS = ',F7.2,1X,'KNOTS')
:     28      CALL EXIT
:     29      END
:STOP
```

Figure 5-13

with FORTRAN and we will explain only the calls to STDS routines. At line 2.25, CALL UNIV ('QQQQ') sets up a universe named 'QQQQ'. At line 3, CALL GET (VGH, 'NEWVGH') retrieves the data base 'NEWVGH' and assigns a local variable VGH for future reference. The cardinality of the set VGH is obtained with the CARD function and then printed by FORMAT 94. At line 4, we make our primary call to STDS. We request that elements having altitude (domain 10) between 8000.0 and 9000.0 feet be placed in a new set called INT. The cardinality of INT is obtained and printed. Beginning at statement 5, we calculate average TAS for the elements of the set INT. FORTRAN statement 100 is the top of the loop that traverses the set INT. Statement 500 is the exit condition executed when the set INT is exhausted.

The loop begins by obtaining the cardinality of INT. If it is non-zero we increment the element counter, N. Then we get the actual values for Q, ALT, and PS in the first element of the set by calling GETELM. Parameters three and four in the call are byte-oriented rather than domain-oriented. This is a human factors disadvantage of a number of the STDS functions. Then we calculate TAS and add the result to the total TAS, TSUM. Then we delete the first element of INT in the following manner. We isolate the first element of the set in the new set UNIT. Then we "subtract" UNIT from INT thus creating INT with one less element. At this point we continue by jumping to the top of the loop. At statement 500, we have completed the task and we print the average TAS.

Figure 5-14 demonstrates how one compiles and executes the program from MTS. The command `RUN -L+ RBLG:STDSSUB* + OLD: LIBRARY` loads the file `-L`. In addition, it loads the STDS subroutines from the file `STDSSUB*` under the user ID "RBLG". Furthermore, those routines require some additional routines that are found on and loaded from the file `LIBRARY` under the ID "OLD". Details of this process may be found in MTS, Volume I [3].

After loading MTS issues the message "EXECUTION BEGINS". At this point our program is in control. We print the cardinality of the data base NEWVGH as requested in line 3.5 of the program. We print the cardinality of INT as requested in line 4.25. Finally, we print the average TAS for the altitude interval 8000.0-9000.0 feet as requested in line 26 of the program. Figure 5-15 gives the time and costs for the session. The total cost of the session was less than \$5.00 (at university rates) and the duration of the session was less than five minutes.

```
#RUN *FTN PAR=S=TAS,L=-L  
#EXECUTION BEGINS  
NNO ERRORS IN MAIN
```

```
#RUN -L+RBLG:STDSSUB*+OLD:LIBRARY  
#EXECUTION BEGINS  
CARD OF NEWVGH = 7232  
CARD OF INT = 39  
AVERAGE TAS = 118.23 KNOTS  
#EXECUTION TERMINATED
```

Figure 5-14

#SIGNOFF		
#OFF AT 01:44.05	10-19-74	
#ELAPSED TIME	4.754 MIN.	\$.23
#CPU TIME USED	29.592 SEC.	\$2.67
#CPU STOR VMI	32.867 PAGE-MIN.	\$1.78
#WAIT STOR VMI	1.336 PAGE-HR.	
#DRUM READS	94	
#APPROX. COST OF THIS RUN IS		\$4.68
#DISK STORAGE	19.25 PAGE-HR.	\$.01
#APPROX. REMAINING BALANCE:		\$257.28

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 5-15

6.0 Concluding Remarks

An information system incorporating the set-theoretic approach (e.g. STDS) provides a flexible environment within which to solve scientific problems needing data base support. The scenarios described herein suggest that the engineer or scientist can manipulate his own data base and obtain reliable answers to his queries in a simple and direct fashion. Furthermore, he may access his data base(s) from either executing programs or from on-line interactive terminals and he may access his data base(s) using named variables rather than storage addresses. In addition, STDS provides a general capability to produce statistical information.

References

1. Hardgrave, W. T., Set processing in a network environment. ICASE Report 75-Hampton, Virginia, March 1974.
2. STDS/OS Users' Guide. Set Theoretic Information Systems, Ann Arbor, Michigan, 1974.
3. The Michigan Terminal System, Volume 1. Wayne State University Computing and Data Processing Center, Detroit, April 1974.

Appendix A

VGH SUMMARY INFORMATION

1. Aircraft serial No.
2. Average true airspeed over entire A/C flight history (total).
3. Total flight miles flown.
4. Total flight hours flown.
5. Number of flights.
6. Average true airspeed during check flight.
7. Average true airspeed during operational flight.
8. Flight miles during check flight.
9. Flight miles during operational flight.
10. Flight hours during check flight.
11. Flight hours during operational flight.
12. Number of check flights
13. Average true airspeed during each flight condition (climb, cruise, descent or operational - x-country).
14. Flight miles during each flight condition.
15. Flight hours during each flight condition.
16. Average TAS during altitude intervals.
17. Flight miles during altitude intervals.
18. Flight hours during altitude intervals.
19. Number of flights during altitude intervals.
20. Percent of flights in time duration intervals (15 min. intervals).
21. Average indicated airspeed - total.
22. Average IAS during check flight.
23. Average IAS during operational flight.

24. Average IAS during each flight condition.
25. Average IAS in rough and smooth air.
26. Average IAS in altitude intervals.
27. Average IAS in type of air during altitude intervals.
28. Percent of flight hours in velocity intervals (or 10, 20 knot intervals).
29. Total average altitude.
30. Average Alt. during check flt.
31. Average Alt. during operational flt.
32. Average Alt. during each flt. condition.
33. Average Alt. in rough and smooth air.
34. Gust, pos. and neg. maneuvers, limit load factors.
35. Occurrences of maneuver accels. in .1 g intervals.
36. Occurrences of maneuver accels in check flight (.1 g intervals).
37. Occurrences of maneuver accels. in operational flight (.1 g intervals).
38. Occurrences of maneuver Accels. in each flight cond. (.1 g intervals).
39. Occurrences of maneuver accels. in each alt. interval.
40. Occurrences of gust accels. in .1 g intervals.
41. Occurrences of gust accels. in each flt. condition.
42. Occurrences of gust accels. in each alt. interval.
43. Occurrences of derived gust velocities (4 FPS intervals).
44. Occurrences of derived gust velocities in each flt. condition.
45. Occurrences of derived gust velocities in each alt. interval.
46. Occurrences of gust accels. during check flt.
47. Occurrences of derived gust velocities during check flt.
48. Reading threshold.
49. Occurrences of maneuver accels in each velocity interval (10 or 20 knots).
50. Occurrences of gust accels in each velocity interval.

51. Occurrences of derived gust velocities in each velocity interval.
52. Flight miles in each velocity interval.
53. Flight hours in each velocity interval.
54. True air speed in velocity interval.